

جامعة الرباط الوطني

National University of AlRIBAT

فرع الخرطوم (المنشية)

Khartoum Branch

برمجة هيكلية

Structural programming

لغة السي

C language

بحث في لغة السي

Search in the c programming language

أعداد الطالب:محمد اسماعيل محمد

اشراف الاستأذ:مصعب

مقدمة(Introduction):

البرمجة الهيكلية:

مبنية على فكرة تقسيم المهام أو المسائل إلى سلسة من المهام ويستمر التقسيم إلى أن نحصل على مهام صغيرة و مستقلة بصورة كافية تمكن من فهمها

لغة السي(C language):

هي لغة برمجة هيكلية للأغراض العامة تحتوى تعليماتها على مصطلحات تشبة التعبيرات الجبرية مدعوة بكلمات محجوزة

نشأة وتطور لغة السي:

قام كل من كين تومسون و دنس ريتشي بتطوير لغة c لبرمجة نظام يونيكس Unix حيث ركزا مطورا هذة اللغة على أن تكون لغتهم سهلة الاستعمال حيث يمكن كتابة برامج كبيرة مع قلة الأخطاء وفى وقت أقصر. في عام 1972 إطلاق لغة c. وكانت مشتقة من لغة ألb وكان نفسها مشتقة من لغة BCPL التي قام بطلاقها مارتن ريتشارد عام 1967 وهى مختصرة من Basic combined programming language حيث كان الفرق بين اللغتين هو نوع البيانات التي قام بتطويرها كين تومسون في عام 1969 حيث اخذ حرفB من اسم المختبرBellحيث كان يعمل في شركة Bell telephone وكان c والذي يلي الحرف Bفي اللغة الإنجليزية هو الحرفC وذلك سبب تسميتها لغة C وفى عام1978 أصدر كل من دنس وبرين (وصفاً لها)قاما بتأليف كتاب عنها

<u>The C Programming Language</u>

والذي يعتبر المرجع الاساسي لها وكان الكتاب معروف بنسخة K&R C (Kernighan&ritche) والسبب في تسمية هكذا أنة بعد أن كثر استعمال لغة السي بشكل كبير والذي أدى إلى تطوير مكتبات ودوال في نسخ مختلفة حتى أصبح كل من تلك النسخ غير متوافقة مع بعضها وهذا أدى إلى تعريف نسخة قياسية للغةc في عام1989تم إطلاق النسخة القياسية للغة السي وسميت بـ ANSI c وهى مختصرة من American National Standards Instituteاي اللجة الوطنية الأمريكية للمعايير وبالتعاون بين اللجنة الوطنية الأمريكية للمعايير والمنظمة الدولية للمعايير تم

إطلاق لغةc قياسية في مختلف أنحاء العالم وسميت بـISOوهي اختصار International Organization for Standizationوكانت مختلفة بعض الشي عن نسخةK&R وللغة Cعدة مميزات:

1- المرونةFlexibility:تحتوي لغة السي على سمات باستخدامها على مستوى منخفض(برمجة نظم التشغيل)
2- كتابة برامج مصدر مجزئة
3- قابلية النقلportability:يمكن ترجمة وتنفيذ البرنامج على مختلف أنواع الأجهزة

البرنامج:

هو كلمة لوصف مجموعة من التعليمات(Source code) كتبت بواسطة المبرمج أو لوصف برمجيات قابلة للتنفيذ (Executable software) ويمر البرنامج بمراحل عدة :

*مرحلة كتابة البرنامج:يتم فيها إنشاء الملف المصدري

الملف المصدري(Source code):

هو سلسة من التعليمات أو الأوامر التي توجة للحاسب الالى لأداء مهمة معينة وكتبت بلغة قريبة من لغة الإنسان

*مرحلة الترجمة(Compiling):ويتم فيها ترجمة البرنامج إلى لغة الآلة

المترجمات(Compilers):

هي عبارة عن برامج تقوم بتحويل الملف المصدري من لغة المستوى العالي إلى لغة الآلة(المستوى الأدنى) منتجة بذلك الملف الغرضى Object file الذي يحتوى على تعليمات مطابقة لتعليمات الملف المصدري

*مرحلة التنفيذ:

الرابطLinker

هو برنامج يعمل على ربط الملف الغرضى للبرنامج مع الملفات العرضية التي تتضمن الدوال المكتبية المستخدمة في توليد الملف التنفيذي Executable file الذي يحتوى على النتائج النهائية للبرنامج

الهيكل العام للغة C:

Header files
Preprocessor Directive
Main Function

انوع البيانات فى لغة (Data type)c:

البينت التى تتعامل معها اما ارقام فاحروف او كلمات

-والارقام يكمن ان تكون صحيحة(ليس بها علامات عشرية)integer او حقيقية(ليس بها علامة عشرية)float

المتغيرات:-

هي معرفات تستخدم لتمثيل بعض أنواع البيانات المحددة داخل جزء محدد من البرنامج لحجز مساحة في الذاكرة ويجب الإعلان عنها قبل ظهورها في عبارات التنفيذ statements والإعلان عن المتغير نكتب نوع البيانات يتبعه اسم المتغير واحد أو أكثر وينتهي بفاصلة منقوطة.

الصيغة العامة syntax:-

```
Data type variable name;
Data type var1, var2, var3,……;
```

إعلان لحجز مساحة في الذاكرة لبيانات صحيحة int

إعلان لحجز مساحة عن متغير حقيقي وعشري float

إعلان لحجز مساحة عن الحروف والعلامات char c

مثال:-

```
#include<stdio.h>
Void main ( )
{
Int x;
Float s;
Char c;
X=19;
F=18.23;
```

```
C=a;
Printf("\n %d %f %c",f,x,c);
Printf("\n %d",x);
```

الثوابت:-

تكون إما أرقام أو سلاسل حرفية ولا يمكن التغير في قيمة الثابت أثناء تنفيذ البرانمج

ويتم تعريفهما بطريقتين:

1- موجه ما قبل الترجمة ويأخذ الصيغة

```
#define const-name value
```

2- أثناء الترجمة يتم إستبدال اسم الثابت بقيمته

```
Const data type valuable name=value;
#include<stdio.h>
#define ch,a ;
#define num 134
Void main( )
Const char 2=,b;
Const in mm2=276
Printf ("%c ", ch );
Printf (" %d",num);
Printf ("%c",ch2);
Printf("%d", num2);
}
```

نوع المتغير	طولة بالبايت	المدى المسموح
حرفى char	1	حرف او رمز واحد
صحيح قصير int	2	32768-32768
صحيح طويل long	4	-2014704830648 2014704830648
حقيقى float	4	E+38-e-38
حقيقىمضاعفdouble	8	E+308-e308

متغير من نوع حرف : أى متغير يصلح لتخزين حرف فقط.

متغير من نوع صحيح : أى متغير يصلح لتخزين رقم صحيح(ليس به علامة عشرية)

متغير من نوع صحيح ولكن طويل(Long): أى يستطيع أى يخزن رقم صحيح ضعف المتغير الصحيح العادى ويستعمل هذا النوع إذا كانت الأرقام التى تتعامل معها أكبر من الم ساحة المخصصة وإلا سنحصل على نتائج خاطئة بالرغم من إن البرنامج سليم

متغير حقيقى : أى متغير يصلح لتخزين رقم حقيقى يقبل الكسور العشرية مثل 6.33

متغير حقيقى مضاعف : أى يستطيع أن يخزن رقم حقيقى ضعف المتغير الحقيقى العادى

- تسميته المتغير – : يخضع اسم المتغير لشروط معينه

-يجب أن يبدأ المتغير بحرف ثم يكمل المتغير بعد ذلك حروف أو أرقام

-يفرق المترجم بين الحروف الصغيرة والكبيرة فالمتغير HPيختلف عن المتغير hp فإذا استعملا فى البرنامج يعتبرهما البرنامج متغيرين

-يجب ألا يكون المتغير بإسم كلمه من الكلمات المحجوزة

٢ –مؤثرات المقارنة Relational operators: وتستخدم لمقارنة قيمتين:

المؤثر	الرمز	مثال	النتيجة
أكبر من greater than	>	100>1	1

1	10<8	<	less thanأصغر من
0	10= =8	= =	equal to يساوى
1	10! =8	!=	not equal to لا يساوى
0	100<=8	<=	less than or أقل من أو يساوى equal to
0	100>=9	>=	greater than or أكبر من أو يساو equal to

o- المؤثرات المنطقيةLogical operator

النتيجة	مثال	الرمز	المؤثر
1	10 > 8 && 9 > 7	&&	و And
0	1 0 < 8 || 7 < 8	||	او Or
1	! (10 = = 8)	!	لا NOt

العوامل الحسابية فى لغةc:

use الاستخدام	Group العامل
Combinationالجمع	+
IPO الطرح	–
Quotient القسمة	/
Beatingsالضرب	*
The rest of باقى القسمة division	%

دوال الإدخال والإخراج input and output

دالة الطباعة على الشاشة printf( )

سوف نقوم بكتابة برنامج زمنة نشرح الادخال والاخرج

#include<stdio.h>

```
void main()
{ printf("hello world");
return 0; }
```

يعد هذا ابسط برنامج يكتب بلغة السى حيث يقوم بترجمة وتنفيذ وطباعة جملة hello world على الشاشة فى بيئة الconsole والسطر الاول بة#include<stdio.h> وهو استدعاء للملف الراسى header file حيث أن ملف الهيدر اسمه هنا هو (stdio.h) أما كلمة include فهى تستخدم لاستدعاء عدد من الملفات منها دالة الزمن (time.h) دوال التعامل مع السلال الحرفية (string.h) وملف يحتوى على جميع الدوال الرياضية (math.h) .

الملف الراسى stdio.h وهو ماخوذ من standard input output وهو امتداد للملف الراسى

ولدينا()main:وهذ ا الجزء مهم جد اً ولا يمكن الإستغناء عنها في أي برنامج للغة السي وهي الد الة الرئيسة للبرنامج

ولدينا العلامتين}و{والتى كل من نهما بداية ونهاية الدالة main

ثم ياتى جزء;("printf("hello world وهذ ا الجزء هوالذي يتو لى طباعة المخرجات على الشاشة حيث أن الد الة (printf) هي الد الة الرئيسة لطباعة شيء ما على شاشة المستخدم وعند إستخد امها لا بد من إستدعاء ملف الهيدر(stdio.h) أم الكلام المحصور بين علامتي التنصيص فهو الكلام الذي سوف يتم طباعته على الشاشة . أما الفاصلة المنقوطة في نهاية السطر فلابد من ذكرها حيث أنه عند عدم ذكره ا سوف يعطيك المفسر رسالة خط أ .والفائدة من الفاصلة المنقوطة أنها تعطي إشارة للمفسر أنه قد تم الإنتهاء من هذ ا السطر ويجب الانتقال للسطر الذي يليه.وهي كما قلنا لا بد أن تكتب حيث أن اغلب الأخطاء تكون منها

(; return 0): وهي تعني أن البرنامج سوف يرجع القيمة الصفرية للدالة (main) حيث أن الدوال في لغة السي يجب أن تعود لها بقيمة إلا إذا كانت هذه الدالة لا تقبل بإعادة قيمة ما

بعض الشروط اللازمة عند كتابة أي برنامج بلغة السي :

-لابد أن يبد أ أي برنامج في لغة السي بإستدعاء م لف الهيدر حيث أنك لا تستطيع أن أن تستعمل الدو ال في برنامجك إلا بعد إستدعاء م لف الهيدرالخاص به ا .ومثال ع لى ذ لك لو إستخدمنا الد الة دون إستدعاء(printf()) م لف الهيدر (studio.h) فإن البرنامج سوف يعطي رسالة خطأ

-لا بد من ذكر الدالة (() main)في جميع البرامج.

-لا بد أن ينتهي كل سطر في جسم البرنامج بفاصلة منقوطة ونعني بجسم البرنامج هو الجزء المحصور بين العلامتين .({ })

إستخدام العلامة (n\) للإنتقال إلى سطر جديد:

تستخدم هذه العلامة لكي تنقل المؤشر من السطر الحالي إلى السطر الذي يلية و المثال التالي يبين طريقة عم لها:

```
#include <stdio.h>
main ()
{
printf("welcome to the world of c\nI hope you enjoy
with it. \n");
return 0;}
```

لاحظ أنه من أن الكود السابق كانت الجم لة في سطر و احد إلا أن بعد تنفيذ البرنامج أصبح الخرج في سطرين وذ لك لإستخد امنا العلامة(n\) ومن. الممكن أن نستخدم أكثر من علامة سطر جديد مثل (n\n\n\) أي عدد السطور الذي تريد المؤشر أن يتخطاه ا.و أيضا يمكن أن تضع هذه العلامة في نهاية النص

```
Printf("welcome to the world of c \n");
```

ويوجد هناك العديد من هذه العلاما ت في لغة السي وهي تسمى بحالات والجدول التالي يبين هذه الحالات

الرمز	الغرض
\n	تنقل المؤشر إلى سطر جديد
\'	هذه العلامة تقوم بطباعة العلامة(') ع لى الشاشة ولاحظ أن أكثر العلامات مثل علامات الإستفاهم وغيرها إذ أردت طباعتها ع لى شاشة المستخدم فلابد أن تكون مسبوقة بالشرطة المائلة و السبب في ذ لك يعود أن أكثر هذه العلامات مستخدمة من قبل لغة السي حيث أنها معرفة في المفسر أنها تقوم بعمل ما.
\"	تقوم بطباعة.(")
\?	تقوم بطباعة(?)
\t	

طباعة قيم المتغيرات على الشاشة:

لطباعة القيم الموجودة بالمتغيرات تستخدم أكواد معينة لتحدد نوع البيانات المراد طباعتها بالدالة

print f ()

```
printf ( " % d " , a ) ;
printf ( " % f " , b ) ;
```

فى هذا ا لمثال عندما يقابل مترجم اللغة العلامة % ينظر إلى الحرف التالى لهذه العلامة . ويعتبر

هذا الحرف توصيف لقيمة موجودة بعد العلامة وكل حرف يحدد تنوع معين من البيانات

والجدول التالى يوضح أكواد طباعة أنواع البيانات

الرمز	الاستخدام	مثال
%d	توصيف لمتغير أو ثابت رقمى صحيح (Signed decimal integer) int	printf (" % d " , -10)
%f	توصيف لمتغير أو ثابت رقمى حقيقى (floating point) float	printf (" % p " , 507)
%c	توصيف لمتغير أو ثابت (حرف واحد) char Single character	printf (" % c " , " a ")
%s	توصيف لعبارة حرفية حرف أو أكثر	printf (" % s " , " is ")

دالة الإدخال العامة Scanf ( )*

هى دالة الإدخال الرئيسية التى تس مح بإدخال جميع أنواع البيانات وهى تأخذ نفس المعاملات التى تأخذها الدالة Print f ( ) وتسمع بادخال البيانات من خلال وحدة الادخال القياسية (keyboard)

```
# include < stdio.h >
main ( )
{
int a , b , c ;
float r , s , t ;
char name [10];
printf ( " \n \n enter your name : " ) ;
scanf ( " % s " , name ) ;
printf ( " a = " ) ;
scanf ( " % d " , & a ) ;
printf ( " b = " ) ;
scanf ( % d " , & b ) ;
printf ( " r = " ) ;
```

```
scanf ( " % f " , &r ) ;
printf ( " s= " ) ;
scanf ( " % f " & s ) ;
printf ( " \n welcome % s " , name ) ;
printf ( " \n \n c = a + b = % d " , a + b ) ;
printf ( " \n \n t = r + s = % d " , r + s ) ;
}
```

-يتم الإعلان عن المتغيرات a , b , c , r , s , t , name

- تطبع الدالة ()printfالرسالة enter your name

-تستقبل الدالة( ) scan f العبارة الحرفية التى يدخلها المستخدم ونصفها فى المتغير ...name كذلك المتغيرات الأخرى

-تستقبل الدالة ( ) scan f فى سطر (scan f ( )(" % d " , &a قيمة صحيحة وتخزنها في المتغيرa

#ماذا يعنى المؤشر# &

&a :تعنى تخزين القيمة الصحيحة فى المكان المخزن عنوانه فى المتغير a بمعنى أن a يشير إلى عنوان المكان الذى تخزن فيه القيمة حيث العلامة & تجعل المتغير يشير إلى عنوان المكان

الناتج-:

```
enter your name : ahmed
a = 5
b = 1٠
r = 2٠
٠s = 3
welcome ahmed
c = a + b = 15
t = r + s = 50
```

دوال إدخال حرف

هناك دوال أخرى تتعامل مع أنواع خاصة من البيانات كالحروف والعبارات الحرفية وهى

putchar()،getchar ( ) , getche ( ) , getch ( )

الدالةgetchar():

تستخدم لادخال حرف واحد ويظهر الحرف على الشاشة بعد الكتابة ولاتسمح بالنتقال الى الامر التالى الا اذا ضغط المستخدم على مفتاح Enter والدالة معرفة داخل المكتبة stdio.h

```
char a;
a=getchar();
printf(“%c”,a”);
```

تم ادخال حرف واسنادة للمتغيرa

الدالة getche():

تستخدم لادخال حرف واحد ويظهلر الحرف على الشاشة ولكنها تختلف عنgetchar() فى انها لاتحتاج الى الضغط على مفتاح Enterللانتقال الى للامر التالى والدالة معرفة داخل المكتبة conio.h()

```
char a;
a=getche();
printf(“%c”,a”);
```

الدالة getch():

تستخدم لطباعة حرف واحد وهذا لايظهر على الشاشة وهى لاتحتاج الى الضغط على مفتاحEnter للانتقال الى للامر التالى وهى معرفة داخل المكتبة conio.h

```
char a;
a=getch();
printf(“%c”,a”);
```

الدالة putcahar

تستخدم لطباعة حرف واحد على الشاشة وهى معرفه فى stdio.h

```
char a;
a=getch();
putchar(a);
```

تطبع الحرف المخزن فى المتغير a

Control Statement عبارات التحكم

تنقسم الى قسمين:

1-عبارات الاختيار Selection:

يتم فيها اختيار تنفيذ عبارة او مجموعة عبارات وفقا لتحقيق شرط محدد

مثل if,if…else,switch

2-الدورات loops:

وفيها يتم تنفيذ عبارة او مجموعة عبارات لعدد من المرات

مثل for,while.do….while

عبارات الاختيار Selection Staements :

{1}عبارة if:

لتنفيذ جملة أو أكثر حسب شرط معين (اختبار منطقى)

الصورة العامة:

```
if ( condition )
statement ;
```

معناه إذا تحقق الشرط (condition) نفذ الجملة التالية أما إذا لم يتحقق الشرط فلا تنفذ هذه الجملة وانتقل إلى التى تليها و إذا كان هناك أكثر من جملة تريد تنفيذها مع if لابد من فتح قوس { قبل مجموعة الجمل والقوس } فى أخر الجمل كما يلى:

```
if ( condition )
{
statement 1 ;
statement 2 ;
}
```

وكمثال:

```
#include <stdio.h>
main()
{
float sum;
```

```
printf("\n Enter the Sum : ");
scanf("%f",sum);
if (sum >50)
printf ("\n The student had passed");
}
```

وفي هذا البرنامج يطبع الكمبيوتر رسالة ليسأل المستخدم عن مجموع الطالب وبعد ذلك يقوم

بمقارنتها بالشرط اللزم للتأكد من النجاح (وهو تجاوز المجموع) 50 فإذا تحقق الشرط يطبع

الكمبيوتر رسالة للمستخدم يعلمه أن الطالب ناجح،

العبارة الشرطية (if .... else statement)

لو نظرنا للبرنامج السابق لوجدنا سؤال ملحا : ماذا لو كان مجموع الطالب أقل من50؟؟؟؟؟؟؟؟؟؟؟

الجابة على هذا السؤال هي أن الطالب يكون راسبا .ولكن البرنامج ل يتضمن أمرا بإعطاء حالة الرسوب، لننا استخدمنا عبارة الشرط البسيطة والتي تستجيب لشرط واحد.

وسنتعرض الن لعبارة مركبة كما في البرنامج التالي:

```
#include <stdio.h>
Main()
{
float sum;
printf("\n Enter the Sum : ");
scanf("%f",sum);
if (sum >50)
printf ("\n The student had passed");
else
printf("\n The student had failed");
}
```

الصورة العامة:

```
if ( condition)
statement-1;
```

```
else
statement-2;
```

حيث أن (condition) هو الشرط

Statement- 1 هي عبارة النتيجة الصلية.

Statement- 2 هي عبارة النتيجة البديلة.

لحالتين القرار اتخاذ من تمكننا - الكاملة الشرطية العبارة باستخدام- وهكذا

لو ماذا والن ، متضادتين

كانت النتيجة الصلية و النتيجة البديلة تتضمنان أكثر من أمر للكمبيوتر؟

أقواس من قوسين بين الصلية النتيجة عبارات احتواء إلى نحتاج الحالة هذه في

وهو البلوكات،

الموضح بالشكل

```
if ( condition)
{
statement 1;
statement 2;
statement n;
}
else
{
statement 1;
statement 2;
statement m;
}
```

والمثال التالي هو البرنامج السابق بعد تعديل

عبارات النتائج لتصبح بلوكات، وذلك ليتمكن البرنامج من إعطاء تقرير بالنجاح أو

الرسوب متضمنا

النسبة المئوية باعتبار المجموع الكلي 1000 في حالة النجاح أو رسالة تفيد بأنه ل

يمكن احتساب

النسبة المئوية لطالب راسب.

لو افترضنا انه قد طلب منك – الرباط- عمل برنامج يمكنه احتساب التقديرات
اعتمادا على مجموع
الطالب، في هذه الحالة نستخدم عبارة شرطية أيضا ولكن بها عدد من الشروط وعدد
مناظر من
النتائج .أو ما يطلق علية العبارة الشرطيةالمتداخلة

```
if ( condition –1)
statement –1;
else if ( condition-2)
statement-2;
else if( condition-3)
statement-3;
………..
else
statement-n;
```

وكمثال:

برنامج عمل الة حاسبة:

```
#include<stdio.h>
#include<conio.h>
Main()
{
Float num1,num2;
Char op;
Printf("\n type num1,op,num2;
scanf("%f%c%f",&num1,&op,&num2);
if(op=='+')
printf("\n sum=%f",num1+num2);
else if(op=='-')
prinf("\n dub=%f",num1-num2);
else if(op=='*')
printf("\n multi=%f",num1*num2);
else if(op=='/')
prinf("\n div=%f",num/num);
```

```
else
printf("ERRoR");
getch();}
```

برنامج قياس العمر

```
#include<stdio.h>
#include<conio.h>
Void main()
{ int dd/mm/yyyy,dd2/mm2/yyyy2;
Printf("the year now:");
Scanf("%d",& dd/mm/yyyy);
Printf("the year you born:");
Scanf("%d",& dd2/mm2/yyyy2);
Printf("you have %d years!\n", dd/mm/yyyy - dd2/mm2/yyyy);}
If(dd/mm==dd1/mm1)
Printf("happy birthday to you");
 }
```

عبارة switch:

يكمن ان تكون بديلة عن if….elseالمتداخلة تتسبب فى تنفيذ مجموعة عبارات معينة من عدد من المجموعات المتاحة للاستخدام ويعتمد الاختيار على القيمة الحالية لتعبي موجود داخل عبارة switchوتاخذ الصيغة:

```
switch (variable)
{
case value1;
statement 1;
break;
case value2;
statement 2;
break;
case value 3;
statement 3;
break;
……………
```

```
default:
statement;
}
```

وكما نرى فإن الختيار المتعدد البدائل يبدأ بكلمة (switch) يليها متغير الختيار والذي تحدد قيمته الختيار الذي سيتم تنفيذه، ويلي ذلك قوس بلوك كبير يحتوي داخله بلوكات صغيرة كل منها يمثل اختيارا من البدائل المطروحة و كل بلوك من بلوكات البدائل يبدأ بكلمة (case) متبوعة بقيمة لمتغير الختيار - والتي تمثل الشرط - وبعد ذلك تأتي عبارة النتيجة.

ويختتم بلوك البديل بكلمة(break) والغرض من هذه الكلمة هو منع الكمبيوتر من تنفيذ عبارة النتيجة التالية وقد تبدو هذه العبارة غريبة للوهلة الولى ويتبادر للذهن سؤال ملح : ألم يتحقق الشرط الول مثل فماذا يدفع الكمبيوتر لتنفيذ بقية عبارات النتائج؟؟

والجابة عن هذا السؤال هي أن عبارة الختيار متعدد البدائل ل ترسل للكمبيوتر أمرا بالتوقف بعد تحقق أي شرط فيها، لذا لزم الستعانة بكلمة(break) وبعد نهاية بلوكات البدائل تأتي كلمة (default) متبوعة بعبارة أو بعبارات ينفذها الكمبيوتر في حالة عدم تحقق أي من الشروط السابقة.

عوامل الزيادة والنقصان(++و--)

أن عامل الزيادة يزيد قيمة معاملة بمقدار واحد وعامل النقصان ينقص معاملة بمقدار واحدة

الصيغة العامة:

- النقصان بمقدار واحد
- ++ الزيادة بمقدار واحد

عبارfor:

هى عبارة احادية اى تحتاج الى معامل تستخدم الحلقة for لتكرار أمر معين (أو مجموعة من الوامر)عددا من المرات وتحتاج الحلقة إلي ثلث عناصر أساسية (انظر الشكل التالي) :

for (counter statement; condition; step)

حيث ان:

-العداد (counter) وظيفة العداد هي تسجيل عدد مرات التكرار

-الشرط (condition) والشرط الذي يحدد نهاية التكرار إذ يظل التكرار قائما حتى ينتفي الشرط.

- 3- الخط(step) وهي القيمة التي تحدد عدد مرات التكرار.

او:

for(exp1;exp2;exp3)

وتقوم for بتكرار اول عبارة بعدها (بسيطة او مركبة) ويستمر التكرار طالما ان ((condition)اوexp2) صحيح

والشكل التالي يوضح برنامجا قمنا فيه باستخدام الحلقfor

```
#include <stdio.h>
main()
{
int counter;
for ( counter=1;counter<=20;counter++)
printf("%d",counter);
}
```

-ومن البرنامج السابق نجد أن الحلقة for بدأت بكلم (for) متبوعة بقوسين بينهما ثلاثة عبارات تفصل بينها علامة الفاصلة المنقوطة العبارة

- الولي تخزن القيمة ابتدائية في العداد والعبارة

- الثانية هي الشرط وهنا الشرط أن قيمة العداد أقل من أو تساوي20

-أما العبارة الثالثة فهي تحدد الخطوة، وفي هذا البرنامج يزاد العداد بمقدار 1 كل مرة تنفذ فيها الحلقة.

-والبرنامج السابق ينتج عنه طباعة الرقام من 1 إلى2

ملاحظة:

-العبارات الثلاثة المكونة لحلقة for يجب أن تفصل عن بعضها بالفاصلة المنقوطة،

-في حالة تكرار أكثر من أمر يتم استبدال العبارة التي تلي بداية الحلقة for

(المثال السابق هي العبارة

printf (" %d",counter;)

forالمتداخلة nested for :

عبار ة عن دورة كبيرة تشمل بداخلها على دورة او اكثر وتاخذ الشكل :

```
For(exp1;exp2;exp3)
 For(exp1;exp2;exp3)
  For(exp1;exp2;exp3)
```

وكمثال ناخذ خدول الضرب من 1-12

```
#include<stdio.h>
main()
{ int i,j;
 for(i=1;i<13;i++)
for(j=i;j<13;j++)
for("%d*%d=%d"I,j,i*j);
}
```

الحلقةwhile loop (while):

تستخدم لتكرار عبارة او مجموعة عبارات stmلعدد معلوم من المرات ويتوقف

التكرار على شرط موجود فى عبارة while

الصيغة العامة"

```
While(exp)
Stm;
```

والمثال الموضح بالشكل التالي يوضح استخدام الحلقة while لطباعة العداد من 1 إلى 2 :

```
#include <stdio.h>
main()
{
int counter=1;
while ( counter <=20 )
{
printf("%d",counter);
```

```
counter++;
}
}
```

من المثال السابق يمكننا استخلص النتائج التالية عن الحلقة:

تخصيص القيمة البتدائية للعداد تتم خارج الحلقة

زيادة العداد تتم داخل الحلقة

الحلقة التكرارية do…..while:

تستخدم لتكرار شرط اومجموعة من عبارات اكثر ن مرة وفقاً لشرط معين مثل while

الصيغة العامة:

```
do
{
statement 1;
statement 2;
.
.
statement n;
}
while ( condition
```

وأهم ملحظة على الحلقة التكراريةdo-while أنها تنفذ العبارات المطلوب تكرارها مرة واحدة على القل حتى ولو كان الشرط غير متحقق

وتفسير ذلك أن التحقق من الشرط يتم بعد التنفيذ وليس قبله كما في الحلقتين السابقتين

يتم تكرار stm طالما ان exp صحيح

يمكن ان تكون stm بسيطة او مركبة

الفرق بين while&do….while:

ان while تختبر الشرط اولا ثم تنفذ العبارة ولكن do..whi.le تنفذ العبارة اولا ثم تختبر الشرط

اى انها على الاقل تنفذ العبارة مرة واحدة

وكمثال:

```
# include<stdio.h>
# include<conio.h>
main( )
{
char pass[10];
do
{
printf("\n enter password: " );
scanf("%s",pass);
}
while(strcmp(pass,"dahe")!=0);
}
```

ملاحظات:

هنا كلمة السر سوف تظهر أثناء الكتابة

الدالة( )strcmp : تقوم بمقارنه متغيرين من نوع عبارة حرفية string فإذا كان المتغيرين متطابقين كان الفرق بينهما صفر

تعديل لبرنامج كلمة السر-:

(عدم ظهور كلمة السر التى يكتبها المستخدم على الشاشة):

```
# include<stdio.h>
# include<conio.h>
main( )
{
chat ch;
char pass[10];
do
{
textcolor(WHITE);
textbackground(BLUE);
cprintf("\n enter password: " );
textbackgrounf(WHITE);
cscanf("%s",pass);
}
while(strcmp(pass,"dahe")!=0);
```

}

الدوال FUNCTION

الدالة عبارة عن برنامج فرعي يودي مهمة محددة ويخصص لها اسم يتم استدعاؤها به او داخل اي دالة اخرى تحتوي مكتبة لغة السي على مجموعة MAINداخل الدالة كما يمكن بناء دالة خاصة غير (printf,scanf,getcharمن الدوال القياسية مثل متوفرة في مكتبة اللغة.

```
#include<filename.h>
Finclion declarations;…………………….(1)
Main()
{
Statements;
Function calling;………………………..(2)
Statements;
}
Function definition()…………………….(3)
{
Statements
}
```

type argument:نوع القيمة(int,float….) التي تعيدها الدالة اللبرنامج المنادي باستخدام الكلمة المحجوزة reurn متبوعة البالقيمة value قد لا تعيد الدالة قيمة و يكون نوع القيمة المعادة void .

function name : اسم الدالة ويقيد بشروط تسمية المعرفات ldentifiers

type arggument1,type argument2:وهي قائمة الوسائط arguments التي تحدد عدد ونوع المتغيرات التي تسمع باستقبال المعلومات المرمات المرسلة من البرنامج المنادى الى الدالة

1- يمسي الاعلان عن الدالة function deelaration وهو اعلان او اخبار المترجم بوجود الدالة

2- استدعاء الدالة function calling او الاتصال بالدالة (استخدام الدالة)

3- تعريف الدالة funetion definition ويتم فيه تحديد التعليمات من خلالها يتم تادية الغرض المحدد

والدوال فى لغة السى تنقسم الى نوعين:

-دوال اللغة:Built in Function وهى الدوال القياسية مثل دالة printf() و scanf()وهى دوال عامة يستطيع اى مبرمج استخدامها

-دوال المستخدم المبتكرة :

وهى الدوال التى من وضع المبرمج

والهدف منها : انه عند تكرار مجموعة من سطور الأوامر اكثر من مرة فى مواضع مختلفة فإن أوامر التكرار لن تكون ذات منفعة . ولذلك يتم كتابة هذه السطور منفصلة عن البرنامج الأساسى

مزايا استخدام الدوال:

- عدم تكرار التعليمات داخل البرنامج : حيث يتم إنشاء الدالة مرو واحدة ثم يتم استدعائها أكثر من مرة عند الحاجة إليها
- باستخدام الدوال يصبح البرنامج أكثر وضوحاً

تتكون الالة من شقين:

1- الإعلان عن الدالة 2- جسم الدالة

```
# include <stdio.h>
#include<conio.h>
void line2(void);
main( )
{
clrscr( )
```

```
line2( )
printf(“ ** Allah the god of all world ** \n “);
line2( )
/* end of main( ) function */
}
void line2(void)
{
int j;
for(j=0;j<=40;j++);
printf( “ * “);
printf(“\n “);
}
```

فى البرنامج السابق أنشأنا دالة بالاسم line2() وقد ظهرت فى ثلاثة مواضع :

الموضع الأول : يسمى الأعلان عن الدالة function declaration يكون ذلك قبل الدالة الرئيسية main( ) كما فى السطر رقم ٣ ونلاحظ الفاصلة المنقوطة فى نهاية الجزء لأنه أعلان.

الموض ع الث انى : داخل الدالة الرئيسية main()ويظهر فى أى مكان داخل الدالة الرئيسية ويسمى function coling أى استدعاء الدالة ويكون بالشكل(line2()كما فى السطر ٧و ٩ وفيه يتم كتابة اسم الدالة فقط بدون نوع وإذا كان لها معاملات نكتب المعاملات.

الموض ع الثالث: يكتب بعد انتهاء الدالة الرئيسيةmain()وهذا الجزء يسمى تعريف الدالة function definition وفيه يتم كتابة محتويات الدالة . وتبدأ فى البرنامج من السطر رقم ١١ باسم الدالة ثم بالقوس { وكانها برنامج ونبدأ كتابة تعليمات الدالة بعد القوس ثم ننتهى بالقوس}

انواع الدوال Function Type:

دوال تعيد قيمة صحيحة	int function

float function	دوال تعيد قيمة حقيقية
string function	دوال تعيد عبارة حرفية
void function	دوال لا تعيد اى قيمة
struct function	دوال تعيد قيمة من نوع structure

```
# include <stdion.h>
int sum(int a, int b )                    الاعلان عن الدالة
main( )
{
int z , x = 10 , y = 40;
z = sum(x,y);
printf("\n\n z = %d " , z );
}
/* الدالة */
int sum(int a , int b )
{
```



```
int s;
s = a + b ;
return s;
}
```

** ملاحظات على البرنامج **

وهى نوع:

- فى السطر رقم ٢ تم الاعلان عن دالة بالاسم sum( ) وسبقت بالكلمة int وهى نوع الدوال وتقابل كلمة void مع ملاحظة وجود متغيرين بين الأقواس وهما معاملات الدالة
- فى السطر رقم ٦ يتم استدعاء الدالة وبين أقواسها المتغيرات x ,y ويستخدمان كمعاملات للدالة)لابد من كتابة معاملات الدالة لأننا أعلنا عنها بهذه الصورة(
- تشمل السطور من ٩ الى ١٤ على جمل الدالة نفسها:-

السطر رقم ٩ نعوض عن المتغير a بالقيمة الموجودة فى المتغير x وهى القيمة ١٠
كذلك نعوض عن المتغير b بالقيمة الموجودة فى المتغير y. وهى 40.

السطر رقم ١٢ نجمع محتويات كلا من المتغير a والمتغير b ونضع النتيجة فى متغير جديد هو s

السطر رقم ١٣ نطلب اعادة محتويات المتغير s الى مكان استدعاء الدالة باستخدام كلمة return

نفهم ان جملة z = sum(x,y) الموجودة بالسطر رقم ٦ تعادل الجملة z = s

ملاحظة هامة : معنى الدالة يتضح من القاعدة التى تقول أن نوع الدالة يتوقف على القيمة المرتجعة من الدالة.

فإذا كانت القيمة المرتجعة int كان نوع الدالة int

وإذا كانت القيمة المرتجعة float كان نوع الدالة float

أما الدالة التى لا تعيد قيمة(الدالة لا تشتمل على جملة return) فتكون من نوع void

استدعاء الدالة:

- يتم استدعاء الدوال اما بمعاملات او بدون معاملات
- تكون الدالة بدون معاملات مثل الدالة void line2(void) اى عدم كتابة قيم بين أقواس الدالة
- برنامج يوضح كيفية استدعاء الدالة بمعاملات:

```
# include < stdio.h>
# include <conio.h>
void line3(int no)
main( )
{
clrscr( )
line3(30);
printf(" ** Allah the god of all world ** \n ");
line3(70);
}
void line3(int no)
```

```
{
int j , no;
for(j = 0 ; j <= no ; j++ )
printf(" * " );
printf("\n");
}
```

ملاحظة : الدالة هنا لها معامل واحد من نوع صحيح وهوno وفى كل مرة يتم ارسال قيمة مختلفة للمعامل وذلك عند استدعاء الدالة.

استدعاء الدالة بمتغيرات:

ممكن استدعاء الدالة بمعاملات من نوع قيم ثابتة موجودة بالبرنامج نفسه وأيضا يمكن ان تكون هذه المعاملات متغيرات تستقبل قيمها من المستخدم او من داخل البرنامج وهذا يفيد فى حالة تغير واختلاف المتغيرات فى كل مرة) إعطاء مرونة فى التعامل مع البرنامج(

برنامج لتحديد الكمية الأكبر من ثلاث كميات صحيحة:

```
# include <stdio.h>
/* determine the largest of three integer quantities */
main( )
{
int a, b, c, d;
/* read the integer quantities */
printf("\n a = ");
scanf( % d " , &a );
printf("\n b = ");
scanf( % d " , &b );
printf("\n c = ");
scanf( % d " , &c );
/* calculate and display the maximum value */
d = maximum( a, b );
printf("\n \n maximum = % d , maximum(c ,d ));
}
/* determine the larger of two integer quatities */
maximum(x ,y )
```

```
int x ,y;
{
int z;
z = (x >= y ) ? x | y;
return(z);
}
```

ملاحظة هامة:

من ضمن أوامر التحكم علامة الاستفهام الشرطية|؟حيث

اختبار test عبارة (x > = y)

؟ عبارة عن سؤالif

| عبارة عن Else

بمعنى إذا كان الاختبار (x >= y) صحيحا يأخذ القيمة x وإذا كانت القيمة غير صحيحة يأخذ القيمةy

مراجع البحث Reference Search

:

1-كتاب البرمجة بلغة الc++& c

2-مقدمة في البرمجة بلغة C

3-سلسلة الشامل لعلوم الحاسوب والالكترونيات

4-ويكيبديا

5-مقدمة عامة على لغةc

6-مقدمة فى البرمجة بلغة c

c-in7-days-7

The C Programming Language-8